BERT4ETH: A Pre-trained Transformer for Ethereum Fraud Detection

Sihao Hu
National University of Singapore
Georgia Institute of Technology
sihaohu@gatech.edu

Zhen Zhang
National University of Singapore
zhen@nus.edu.sg

Bingqiao Luo
National University of Singapore
luo.bingqiao@u.nus.edu

Shengliang Lu
National University of Singapore
lusl@nus.edu.sg

Bingsheng He
National University of Singapore
hebs@comp.nus.edu.sg

Ling Liu
Georgia Institute of Technology
ling.liu@cc.gatech.edu

ABSTRACT

As various forms of fraud proliferate on Ethereum, it is imperative to safeguard against these malicious activities to protect susceptible users from being victimized. While current studies solely rely on graph-based fraud detection approaches, it is argued that they may not be well-suited for dealing with highly repetitive, skew-distributed and heterogeneous Ethereum transactions. To address these challenges, we propose BERT4ETH, a universal pre-trained Transformer encoder that serves as an account representation extractor for detecting various fraud behaviors on Ethereum. BERT4ETH features the superior modeling capability of Transformer to capture the dynamic sequential patterns inherent in Ethereum transactions, and addresses the challenges of pre-training a BERT model for Ethereum with three practical and effective strategies, namely repetitiveness reduction, skew alleviation and heterogeneity modeling. Our empirical evaluation demonstrates that BERT4ETH outperforms state-of-the-art methods with significant enhancements in terms of the phishing account detection and de-anonymization tasks. The code for BERT4ETH is available at: https://github.com/git-disl/BERT4ETH.

ACM Reference Format:
Sihao Hu, Zhen Zhang, Bingqiao Luo, Shengliang Lu, Bingsheng He, and Ling Liu. 2023. BERT4ETH: A Pre-trained Transformer for Ethereum Fraud Detection. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30-May 4, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3543507.3583345

1 INTRODUCTION

As a decentralized computing platform, Ethereum empowers its developers to create a variety of decentralized applications (DApps). Despite the substantial engagement garnered within the cryptocurrency sphere, Ethereum has also become a hub for a wide range of fraudulent activities, such as phishing scams [31], pump-and-dump schemes [15], Ponzi schemes [6], ICO scams [3], money laundering [30], and bot arbitrage [9], *etc.*

Many recent studies [2, 4, 20–22, 31] employ graph representation learning techniques for fraud detection on Ethereum. Although it is intuitive to represent the interactions between accounts as a graph, it is argued that they have the following limitations:

(i) Graph is not appropriate for capturing the sequential pattern inherent in transactions. Ethereum transactions are high repetitive, indicating the presence of multi-edges between nodes. Current methods [20, 24, 25, 31] integrate multi-edges to a single edge to facilitate graph computations. However, the discarded sequential information is essential for characterizing user behaviors for tasks such as de-anonymization. (ii) Graph-based approaches are ill-suited for capturing dynamic status of accounts as they treat accounts as fixed nodes with fixed features. However, the status of an account changes after a single transaction in Ethereum. (iii) Current methods exclusively target on a single fraud detection task. In light of the successes of Transformer pre-training techniques in NLP [10, 28], we believe that a pre-trained Transformer can support various fraud detection tasks with minimal adaptations needed.

To address the limitations discussed above, we introduce a pre-trained model that offers a universal solution for various fraud detection tasks on Ethereum. BERT4ETH features the superior sequential modeling capability of Transformer [28] and the pre-training paradigm of BERT [10]. In this paper, we first present the architecture of BERT4ETH, with a specific focus on the integration of Transformer into the Ethereum contenxt. BERT4ETH serves as a sequential encoder, capable of extracting representation vectors for user accounts based on their transaction histories. Second, we introduce the Masked Address Prediction (MAP) task, which involves randomly masking addresses (accounts) in transaction sequences and requiring the model to predict the masked addresses. The MAP task forces the model to learn the relationship between addresses (accounts) in transaction sequences.

However, three characteristics of Ethereum pose challenges for pre-training: (i) *Repetitiveness*: High repetitiveness prevents BERT4ETH from learning meaningful representations through the MAP task, because label information is very likely leaked from unmasked addresses to masked addresses. (ii) *Skewed distribution*: The frequency of occurrence of addresses follows the power-law distribution, with a small number of popular addresses proliferating in the majority of transaction sequences. This reduces the distinctiveness of representations that is what fraud detection covet most. (iii) *Heterogeneity*: Ethereum transactions include various types of interactions (Ether/token transfer, contract call) between different types

WWW '23, April 30-May 4, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04.
https://doi.org/10.1145/3543507.3583345

of accounts, creating a challenge in modeling the heterogeneity and uncovering meaningful patterns behind transactions.

We tackle the above challenges by equipping BERT4ETH with the following three strategies:

- *Repetitiveness reduction:* To counter the label leakage problem in pre-training, we first aggregate continuously repetitive transactions while preserving the sequential order. Second, we propose two alternative effective strategies: adopting a *high* masking ratio (80%) or a *high* drop out ratio (80%) during pre-training. These tactics create a task that cannot be easily extrapolated by the high repetitiveness.
- *Skew alleviation:* We emphasize the distinctiveness by sampling high-frequency addresses as negative samples in a contrastive loss function adopted for pre-training. Optimizing the contrastive loss equals to alleviating the negative impact of high-frequency addresses. Additionally, we propose an intra-batch sharing strategy for negative samples, which allows an extremely high negative-to-positive ratio to alleviate the skewness, and decreases the overlap of negative sets because of frequency-aware sampling.
- *Modeling heterogeneity:* BERT4ETH captures fine-grained transaction information by the embedding technique and models sequential pattern by a Transformer encoder. Furthermore, (i) we separate transaction sequences into in/out-type sub-sequences and model them individually because fraudsters can be more easily distinguished in specific types of in- or out-transactions; (ii) We propose a log encoder to integrate the trace of ERC-20 token transfer without introducing noise, since token transfer trace cannot be captured by normal transactions.

Extensive experiments conducted on two crucial fraud detection tasks show that BERT4ETH significantly advances the state-of-the-art performance, achieving a F_1 improvement of **21.61** absolute percentage (AP) for phishing detection and Hit Ratio@1 improvement of **13.54** and **21.57** AP for de-anonymization on the ENS and Tornado (0.1ETH) datasets, respectively.

Contributions: To summarize, the contributions are as follows:

- We present BERT4ETH, a pre-trained Transformer that provides a universal solution for various Ethereum fraud detection tasks.
- We equip BERT4ETH with three effective strategies to generate robust and expressive representations, given repetitive, skew-distributed and heterogeneous Ethereum transaction.
- BERT4ETH significantly advances the state-of-the-arts on two important fraud detection tasks. As a side contribution, we make available the code and dataset.

2 RELATED WORK AND BACKGROUND

2.1 Ethereum Representation Learning

Previous studies have primarily focused on graph-based methods for Ethereum account representation learning, which can be classified as DeepWalk-based and GNN-based methods.

DeepWalk-based method: Trans2Vec [31] is proposed for the phishing account detection task, which integrates temporal and amount information of transactions into its random walk process, making the proximity of learned node representations reflects the relationship between accounts. Other works, such as [21, 22], also take inspiration from Trans2Vec. For the task of de-anonymization, which aims to identify two accounts belonging to a single user based on the proximity of account representations, Beres et al.[2] evaluate 11 graph learning methods on ground-truth pairs collected from the ENS and Tornado coin-mixers. Among them, Diff2Vec[25] and Role2Vec [1] are considered the state-of-the-art methods.

GNN-based method: Shen et al. [26] utilize Graph Convolution Network (GCN) [17] to classify accounts into "normal," "phisher," and "bot" categories based on inferred identities. Zhou et al. [34] propose HGATE, a hierarchical graph attention encoder that integrates features from both node-level and subgraph-level to enhance phishing detection performance. Li et al. [20] propose TTAGNN, a GNN that fuses multiple temporal edges by using a LSTM network, and learns node embeddings through a Graph Attention Network (GAT) [29]. A graph auto-encoder is employed to generate a self-supervised signal for representation learning, with a LightGBM model adopted for the phishing account detection task.

2.2 Transformer & BERT

Transformer [28] is a sequence-to-sequence machine translation model that introduce the groundbreaking self-attention mechanism for capturing the relationship between word tokens. BERT [10] proposes Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) task to pre-train the Transformer encoder in a bidirectional context. It advances the state-of-the-art on eleven NLP tasks with a significant enhancement, and has inspired a number of variants such as ALBERT [18], RoBERTa [23] and XLNet [32]. Moreover, the pre-training paradigm of BERT makes it can be easily extended to various downstream tasks. In light of the success of Transformer and BERT pre-training, our research aims to take advantages of their superior capabilities for Ethereum fraud detection.

2.3 Terminology

Externally Owned Account (EOA): EOAs are accounts controlled by users who own their private keys, allowing them to initiate external transactions for transferring cryptocurrency or triggerring smart contracts. This study focuses on modeling the transactions initiated by EOAs as they are under human control.

Contract Account: Contract accounts are self-executing computer programs deployed on the Ethereum network. Ethereum allows encoding of arbitrary contract functionality. While contract accounts cannot issue external transactions, they can initiate internal transactions.

External Transaction: An external transactions initiated exclusively by an EOA, either transfers cryptocurrency to other accounts or call a contract account to trigger its execuation. In comparison, there are internal transactions initiated by smart contracts to execute complex logic. In this study, the term "transaction" specifically refers to external transactions, unless specified otherwise.

Token: Tokens are digital assets that can be programmed to serve various functions, such as functioning as as a currency, granting access, voting, providing identity and utility. Currently, the majority of tokens are built upon the ERC-20 standard [5].

3 MOTIVATION

We introduce three challenges/characteristics of Ethereum that motivate us to design a new BERT-based model.

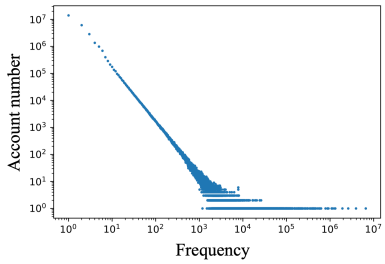

Figure 1: The frequency of occurrence of account (address) follows a power-law distribution.

Repetitiveness: Ethereum transactions are highly repetitive. Statistics indicate that there are 48.4% of transactions share the same receiver with its one prior transaction initiated by the same sender, suggesting that Ethereum users exhibit a tendency to repeatedly interact with the same accounts. However, the pre-training task of BERT4ETH is vulnerable to high repetitiveness: label information can leak from unmasked tokens to masked but repetitive ones, hindering BERT4ETH to capture meaningful co-occurrences between addresses, a phenomenon we refer to as the *label leakage* problem.

Skew-distributed: As shown in Figure 1, the frequency of occurrence of accounts (addresses) follows the power-law distribution [19, 33], which means a small number of high-frequency accounts proliferate in the majority of transactions. This characteristic presents the difficulty for representation learning, as it can diminish the distinctiveness of representations: two accounts that interacted with popular accounts like Uniswap are likely be closely located in the latent space, even if they are completely irrelevant.

Heterogeneous: In Ethereum, there exist different types of accounts, transactions and functionalities associated with calling contract accounts. As compared to human languages, the heterogeneity present in transactions makes it more challenging to discern meaningful patterns and determine the most important elements of information. Given that various downstream fraud detection tasks depend on different aspects of information, we aims to preserve heterogeneity as much as possible during the pre-training phase.

4 BERT4ETH

In this section, we present the design of BERT4ETH, along with three strategies aimed at addressing the above-mentioned challenges, dubbed Repetitiveness Reduction (RR), Skew Alleviation (SA) and Modeling Heterogeneity (MH).

4.1 Transaction Sequence

4.1.1 ***Data Collection.*** We deployed an Ethereum node by Geth and utilized Ethereum-ETL to extract structured tabular data from archived raw data. The table schema used in this paper is at https://ethereum-etl.readthedocs.io/en/latest/schema, where *transaction.csv* is the external transaction file and *trace.csv* is the log file.

4.1.2 ***Sequence Generation.*** For an EOA with address A_0, we collect all the transactions which it was either the initiator or the receiver, and sort transactions in *descending* order based on their timestamps. For each transaction, we collect four features, *i.e.*, address, timestamp and amount, account type, and in/out type. In/out type feature indicates whether the transaction is received or initiated by A_0, account type indicates whether the account is EOA or contract, amount is the value of transferred amount, and timestamp denotes the transaction time. Subsequently, we insert a dummy self-transaction at the head of the sequence. The address feature of the self-transaction is set to A_0 (self-address), and all the other features are set to "Null", to differentiate it from normal transactions.

Transaction De-duplication (RR#1): The first strategy of repetitiveness reduction is transaction de-duplication, aiming to reduce continuous repetitiveness. Continuous repetitiveness refers to transactions that interacts with the same addresses continuously in a sequence. First, we eliminate failed transactions as user may initiate several failed transactions before a final one successfully executed. Second, we aggregate continuous repetitive transactions that have same address, same in/out type and initiated within 72 hours into one, by summing up their transaction amounts and tracking the number of transactions. The timestamp of the aggregated transaction is set to the first timestamp of the original transactions. By adopting de-duplication strategy, we lower the repetitiveness ratio from 48.0% to 14.3%, while still preserving the order of the original sequence.

4.2 Model Architecture

4.2.1 ***Embedding Layer.*** As illustrated in in Figure 2, seven features are generated for each transaction, including address, account type, in/out type, amount, count, timestamp and position. Since amount and count are no-categorical features, we use binning to categorize them. The position index is ranked from 0 to $N-1$.

First, we adopt the embedding technique to encode features to make the model aware of the transaction information. Specifically, for the i-th transaction in the sequence, its transactions features are passed through embedding layers to generate the corresponding feature embeddings, which are then summed to obtain its initial transaction representation $\boldsymbol{h}_i^{(0)} \in \mathbb{R}^d$. Next, we stack the initial transaction representations to form a matrix $\boldsymbol{H}^{(0)} = [\boldsymbol{h}_0^{(0)}, \boldsymbol{h}_1^{(0)}, ..., \boldsymbol{h}_{N-1}^{(0)}] \in \mathbb{R}^{N\times d}$ that encompasses all the information of the transaction sequence.

4.2.2 ***Transformer Encoder.*** For a sequence, BERT4ETH takes $\boldsymbol{H}^{(0)}$ as the input, and passes it through the Transformer encoder consisting of L Transformer layers. Each Transformer layer contains two sub-layers, an Attention sub-layer and a Position-wise Feed-Forward sub-layer. We formalize a Transformer layer as follows:

$$\boldsymbol{H}^{(l)} = \text{Attention}\left(\boldsymbol{H}^{(l)}\boldsymbol{W}_Q^{(l)}, \boldsymbol{H}^{(l)}\boldsymbol{W}_K^{(l)}, \boldsymbol{H}^{(l)}\boldsymbol{W}_V^{(l)}\right) \tag{1}$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V \tag{2}$$

$$\boldsymbol{H}^{(l+1)} = [\text{FFN}(\boldsymbol{h}_1^{(l)}); \cdots; \text{FFN}(\boldsymbol{h}_t^{(l)})] \tag{3}$$

$$\text{FFN}(\boldsymbol{x}) = \text{GELU}(\boldsymbol{x}\boldsymbol{W}_1^{(l)} + \boldsymbol{b}_1^{(l)})\boldsymbol{W}_2^{(l)} + \boldsymbol{b}_2^{(l)} \tag{4}$$

where the projection matrices $\boldsymbol{W}_Q^{(l)}, \boldsymbol{W}_K^{(l)}, \boldsymbol{W}_V^{(l)}, \boldsymbol{W}_1^{(l)}, \boldsymbol{W}_2^{(l)} \in \mathbb{R}^{d\times d}$, and bias vectors $\boldsymbol{b}_1^{(l)}$ and $\boldsymbol{b}_2^{(l)} \in \mathbb{R}^{d\times 1}$ are trainable parameters for the l-th Transformer layer. To facilitate description, we omit the multi-head [10], ResNet [13] and batch normalization [16], but they are adopted in practice.

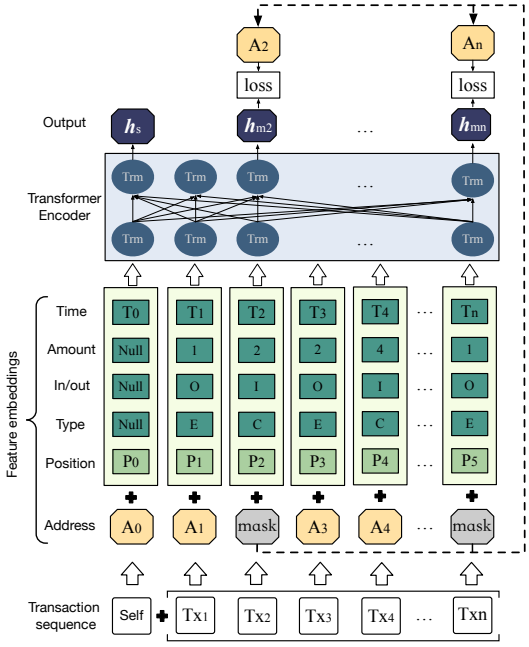

Figure 2: The framework of BERT4ETH pre-training. After a transaction sequence is generated, we select a portion of transactions to replace their addresses with [MASK] and feed the sequence to the model to predict masked addresses.

After L-layer successive calculation, the Transformer encoder produces a matrix $\boldsymbol{H}^{(L)} = [\boldsymbol{h}_0^{(L)}, \boldsymbol{h}_1^{(L)}, ..., \boldsymbol{h}_{N-1}^{(L)}] \in \mathbb{R}^{N \times d}$, where $\boldsymbol{h}_i^{(L)}$ is the final representation of the i-th transaction, which encodes not only its own information but also the bi-directional context information.

4.3 Pre-training

4.3.1 ***Masked Address Prediction***. MAP is derived from the Masked Language Modeling (MLM) of BERT, which involves a *Cloze* test that requires the model to predict the masked addresses in a transaction sequence, as shown in Figure 2. In BERT4ETH, a certain percentage (k%) of transactions within the sequence are selected and their addresses are replaced with the special token [MASK]. The masked sequence is then passed through the embedding and Transformer layers as described before. For a masked transaction, its final transaction representation $\boldsymbol{h}_m^{(L)}$, which encodes its bidirectional contextual information, is used to predict its masked address.

The original BERT predicts masked word tokens through the calculation of probabilities across all tokens (around 30K in number). However, when applied to Ethereum, it is infeasible to calculate softmax$(\cdot)$ across all the addresses as there are up to billions of addresses in Ethereum. Therefore, we adopt a contrastive loss calculated over a positive address and random sampled negative addresses as the objective function for pre-training:

$$L = -\frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \log \left(\frac{\exp(\boldsymbol{h}_m^{\mathrm{T}} \cdot \boldsymbol{a}_p)}{\exp(\boldsymbol{h}_m^{\mathrm{T}} \cdot \boldsymbol{a}_p) + \sum_{n \in \mathcal{N}} \exp(\boldsymbol{h}_m^{\mathrm{T}} \cdot \boldsymbol{a}_n)} \right) \quad (5)$$

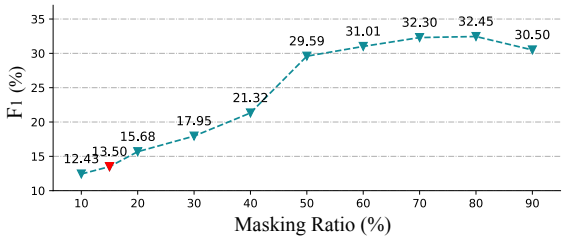

Figure 3: Testing F_1 of phishing account detection *w.r.t.* different masking ratios. A high masking ratio (80%) works significantly better than the original ratio (15%).

where $\mathcal{M}$ is the set of masked addresses in a sequence, $\boldsymbol{h}_m$ is $\boldsymbol{h}_m^{(L)}$ that encodes unmasked contextual information. For each sequence, we samples a negative set $\mathcal{N}$. $\boldsymbol{a}_p$ is the address embedding of its masked address, which we refer to the positive embedding, and $\boldsymbol{a}_n$ is the address embedding of a negative address from the negative set $\mathcal{N}$. Here we reuse the address embedding layer to prevent introducing new parameters. Optimizing Eq. 5 is essentially equivalent to encouraging $\boldsymbol{h}_m$ be closer to $\boldsymbol{a}_p$, and away from $\boldsymbol{a}_n$ in the hidden space.

4.3.2 ***Repetitiveness Reduction***. The issue of high repetitiveness poses a risk of label leakage for the MAP task, which can negatively impact the effectiveness of pre-training. For instance, if BERT4ETH follows the original masking ratio (15%) and uses 85% unmasked addresses to predict 15% masked addresses, the masked addresses have a high likelihood of being present in the unmasked addresses, leading to an overly easy prediction task [12]. This issue, in turn, results in small values for loss and gradients, causing the parameters to be inadequately trained and impeding the model in capturing the meaningful occurrence patterns between addresses.

Despite the proposal of the de-duplication strategy to reduce continuous repetitiveness, the left discontinuous repetitiveness still remains substantial. To mitigate the adverse effects of high repetitiveness, we put forth two effective strategies without introducing any additional operations:

High Masking Ratio (RR#2): A straightforward solution is to increase the masking ratio k to a very *high* value, thereby creating a task that cannot be easily extrapolated. Figure 3 demonstrates the testing F_1 of BERT4ETH on the phishing account detection task (with fixed-training strategy will be described in Section 5.3) by switching the masking ratio from 10% to 90%. Accordingly, the F_1 score increases from 0.1350 to 0.3245, causing a huge performance gap up to **18.95** AP. Among them, k=80% achieves superior performance, and when k>80%, we observe that the performance starts to decrease because unmasked information is too limited for the task.

High Dropout Ratio (RR#3): An alternative approach is to adopt a *high* dropout rate for pre-training, which shares the same idea with raising the masking ratio. Empirical results show that with a low masking ratio of 15%, a similar performance can be reached by adopting a high dropout ratio of 80%. However, the benefits brought by increasing dropout and masking ratio are not cumulative because they achieve the same effect. Therefore, given a masking ratio of 80% adopted, a dropout ratio of 20% is set as the default value based on empirical hyper-parameter tuning.

Table 1: Testing F_1 of phishing detection *w.r.t.* different skew alleviation strategies. † denotes intra-batch sharing.

P/N Ratio	1:20	1:1000†	1:5000†	1:10000†
Uniform	0.3245	0.3554	0.3804	0.3746
Freq(0.5)	0.3313	0.3939	0.4214	0.4203
Freq(1.0)	0.3770	0.4390	0.4365	0.4371
Zipfan	0.4239	0.4251	**0.5044**	0.5036

4.3.3 ***Skew Alleviation***. As previously shown in Figure 1, the occurrence frequency of Ethereum accounts follows a power-law distribution, meaning that a small number of popular accounts are highly likely exists in the majority of transaction sequences, causing two irrelevant accounts be close to each other in the hidden space simply because they interact with the same popular accounts.

A good encoder is expected to identify rare activities out of the majority of transactions, as transactions that interact with low-frequency addresses could be important signals for fraud detection. We present two strategies to alleviate the negative impact of skewed distribution:

Frequency-aware Negative Sampling (SA#1): Given that the masking ratio is high (80%), compared to low-frequency addresses, high-frequency addresses are more likely to be masked and their address embeddings will be selected as $\boldsymbol{a}_p$ for Eq. 5. Optimizing Eq. 5 encourages $\boldsymbol{h}_m$ that encodes unmasked transaction sequence be closer to $\boldsymbol{a}_p$ in the hidden space. As a result, addresses that co-occurr with high-frequency addresses become closer to them, and thus the sequence representations also becomes closer, which is undesired for fraud detection. An effective solution is to take high-frequency addresses as negative samples to counteract the impact of these addresses being trained frequently as positive samples. Specifically, we introduce two frequency-aware sampling strategies: Zipfan sampling and Frequent sampling, as follows:

- Zipfan sampling:

$$P_{neg}(A_i) = \frac{log(r(A_i)+2) - log(r(A_i)+1)}{log(max+1)} \tag{6}$$

  where $r(\cdot)$ is the rank of A_i based on the descending frequency.
- Frequent sampling:

$$P_{neg}(A_i) = \frac{f(A_i)^b}{\sum_j f(A_j)^b} \tag{7}$$

  where $f(\cdot)$ is the frequency of account/address A_i and b is an adjustable hyper-parameter. In the experiment, we set b=0.5 and 1.0. If b=0, it degrades into the uniform sampling.

Intra-batch Sharing (SA#2): Given that we sample high-frequency addresses as negative samples, the negative sets of different transaction sequences would be highly overlapped because they all concentrate on sampling high-frequency addresses. To reduce this waste, we force masked transactions in the same batch to share all the negative samples. Another advantage of this strategy is that given the whole number of negative samples unchanged, the negative/positive ratio largely increases from $|\mathcal{N}|:1$ to $B\cdot|\mathcal{N}|:1$, providing a greater degree of skew alleviation, where $|\mathcal{N}|$ is the size of negative set and B is the batch size for transaction sequence.

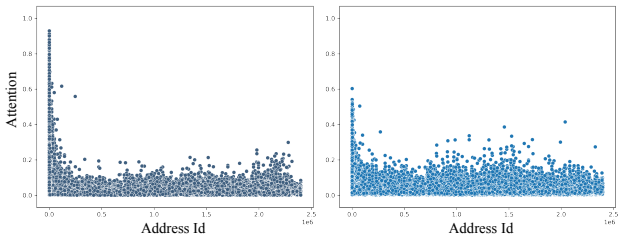

Figure 4: Attention distribution with uniform(a) or Zipfan(b) negative sampling. Address Id is ranked according to the descending frequency.

Table 1 presents the results of skew alleviation strategies on the phishing account detection task. It is obvious that: 1) BERT4ETH achieves better performance when the degree of frequent negative sampling increases (Zipfan> freq(1.0)> freq(0.5)> uniform), and the F_1 gap is up to **9.94** AP; 2) For Zipfan sampling, when the negative-to-positive ratio increases from 20 (without intra-batch sharing) to 1,000, 5,000 and 10,000 (with intra-batch sharing), F_1 increases to 0.5044, and the gap is up to **8.05** AP. As a result, the negative-to-positive ratio of 5,000 is adopted as the default setting.

Figure 4 demonstrate two attention distributions received by addresses in the first Transformer layer. It is obvious that attention scores assigned to the high-frequency addresses are decreased after Zipfan sampling, enabling BERT4ETH pay more attention to low-frequency addresses to enhance the distinctiveness.

4.4 Advanced Features

We designate the above-described model as the basic BERT4ETH. In this section, we present two advanced techniques aimed at tackling the heterogeneity of Ethereum transactions.

In/Out Separation (MH#1): It is observed that fraudulent EOAs exhibit special patterns of in-type and out-type transactions. For phishing EOAs, the in-to-out ratio is 1.250, whereas, it is only 0.385 for normal accounts. The difference arises from the nature of fraud activities: a phishing EOA receives transfers from its victims, thus making the in-type transactions dominant, whereas the out-type transactions might indicate the flow of stolen funds to accomplices or other accounts controlled by the attacker. Due to MAP is a self-supervised task, no fraud label is available to highlight these crucial yet minority transactions. As a result, these transactions may be overlooked during the self-attention computation process.

A solution is to generate other two sub-sequences by separating the original sequence based on transaction's in/out type feature. As a result, another two Transformer encoders are employed to generate $\boldsymbol{H}_{in}^{(L)}$ and $\boldsymbol{H}_{out}^{(L)}$ correspondingly. Self-attention computations are confined within each sub-sequence. Parameters in embedding layers are shared to prevent largely increase of parameter numbers, as the address embedding layer is very large in size.

ERC-20 Transfer Log Encoder (MH#2): Token transfer is implemented at the contract-level rather than at the protocol level. Therefore, a token transfer to a receiver is not recorded by an external transaction. To capture the transfer relationship, we analyze ERC-20 transfer logs from internal transactions, and select the transfer behaviors happened between EOAs to prevent noise.

For an external transaction that invokes ERC-20 token transfer, we associate all the EOAs that receive ERC-20 tokens to the recipient contract address. It should be noted that the number of recipient EOAs is uncertain as a single transaction may transfer tokens to multiple EOAs simultaneously.

Before passing through the Transformer encoder, we first mean pool the address embeddings of attached EOAs to encode their information. Second, we employ a gate mechanism to integrate the embeddings of contract account and recipient EOAs:

$$\boldsymbol{a}_u = \text{Mean}(\boldsymbol{a}_{u1}, \boldsymbol{a}_{u2}, ..., \boldsymbol{a}_{un}) \tag{8}$$

$$\boldsymbol{a}_c^{'} = \boldsymbol{\beta} \cdot \boldsymbol{a}_c + (\mathbf{1} - \boldsymbol{\beta}) \cdot \boldsymbol{a}_u \tag{9}$$

$$\boldsymbol{\beta} = \text{Sigmoid}(\boldsymbol{W}_\beta \cdot [\boldsymbol{a}_c || \boldsymbol{a}_u] + \boldsymbol{b}_\beta) \tag{10}$$

where $\boldsymbol{a}_c$ is the address embedding of recipient contract, $\boldsymbol{\beta}$ is the gate vector that is adaptive to $\boldsymbol{a}_c$ and $\boldsymbol{a}_u$. $\mathbf{W}_\beta \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_\beta \in \mathbb{R}^d$ are parameters optimized during training. Address embedding $\boldsymbol{a}_c^{'}$ is encoded into the initial transaction embedding $\boldsymbol{h}_c^{(0)}$. The gate mechanism prevents introducing noise in instances where transfer actions, such as token airdrops, do not necessarily guarantee the relationship between the initiator and receiver.

4.5 Apply to Downstream Tasks

Pre-trained BERT4ETH functions as a representation extractor for a given transaction sequence. To represent the entire sequence, we pick the representation of the self-transaction $\boldsymbol{h}_s^{(L)}$ because it encodes the global information. In the case where BERT4ETH adopts the in/out separation strategy, the final representation is obtained by concatenating the three representations of self-transactions extracted from $\boldsymbol{H}^{(L)}$, $\boldsymbol{H}_{in}^{(L)}$, and $\boldsymbol{H}_{out}^{(L)}$. If the sequence exceeds the maximum length, it is split into multiple sequences and multiple representations are generated, which are then mean-pooled to produce the final representation.

5 EXPERIMENTS

5.1 Task Description

5.1.1 ***Phishing Account Detection***. Phishing attack is the most prevalent form of fraud on Ethereum. Attackers send victims fake airdrop messages through emails or social networks to lure them into logging accounts on phishing websites [7] or transferring cryptocurrency to the designated phishing accounts [8]. Unlike conventional phishing scams, Ethereum transaction are publicly accessible, allowing us to identify phishing accounts and alert susceptible users before they being victimized. In our experiment, the task of phishing account detection is framed as a binary classification problem, where the goal is to determine whether an EOA is a phishing account. We adopt Precision, Recall and F_1 as the metrics.

5.1.2 ***De-anonymization***. De-anonymization aims to identify two different EOAs controlled by the same user. One application of de-anonymization is to trace the flow of money laundering. For example, Tornado Cash [2, 27] provides coin-mixing services on Ethereum: a participant deposits certain amounts of ether into a Tornado mixer contract, and use another account to withdraw the deposited coins after a period of time. In our experiment, given a ground-truth pair of EOAs, we use the representation of the query EOA to query its top-k closest neighbors in the hidden space. If the target EOA is present among them, de-anonymization is considered as successful. We adopt Hit Ratio@k (HR@k) to measure the performance and use Euclidean distance as the metric for proximity because it yield slightly better results than cosine similarity.

Table 2: Dataset statistics

Dataset	Phishing	ENS	Tornado	Normal
# EOA	3,220	1,335	2,301	594,038
# All address	151,415	57,526	139,178	2,609,855
# Tranx	328,261	821,140	1,056,674	11,350,640
# In tranx	182,356	60,258	150,157	3,159,707
# Out tranx	145,905	760,882	906,517	8,190,933

5.2 Experimental Setup

5.2.1 ***Dataset:*** For phishing account detection, we collect 7,057 accounts labeled by Etherscan. Among them, 97% are EOAs. For de-anonymization, we use ground-truth pairs collected by Xu et al. [2] from two sources: Ethereum Name Service (ENS) and Tornado Cash. For pre-training, we randomly sample 1,000,000 EOAs and filter out accounts labeled as "phishing", "exchange", "miner" and "mining pool". These normal EOAs are also used for negative samples for phishing account detection and candidate set for de-anonymization on ENS dataset. We collect all the transactions that these EOAs has involved in, covering the period from Jan.1 2017 to May.1 2022. We filter out EOAs that has less than 3 transactions or more than 10,000 transactions. The statistics of dataset is presented in Table 2.

5.2.2 ***Baselines:*** We compare three types of baselines in the experiment: 1) DeepWalk-based methods, including DeepWalk [24], Trans2Vec [31], Diff2Vec [25] and Role2Vec [1]. Trans2Vec is specifically designed for phishing account detection. Diff2Vec is the SOTA method for de-anonymization in [2]; 2) GNN-based methods, including GCN [17], GraphSAGE [11] and GAT [29]; 3) BERT4ETH and its variants, including the basic BERT4ETH, BERT4ETH† with in/out separation and BERT4ETH§ with ERC-20 log encoder.

5.2.3 ***Implementation:*** For BERT4ETH, the number of Transformer layers is set to 8, number of attention head is set to 2 and maximum sequence length is set to 100. For graph-based methods, we adopt the self-supervised training task proposed in DeepWalk [24]. We set the number of walks per node to 10, walk length to 20, and context size to 5. For GNN-based methods, the number of GNN layers is set to 2 with the neighbor sample size of 50. For all the competitors, the negative-to-positive ratio is set to 20, hidden dimension set to 64, batch size set to 256 and dropout ratio set to 20%. Other parameters of BERT4ETH follows the previously mentioned default settings.

5.3 Performance Comparison

5.3.1 ***Phishing Account Detection:*** We evaluate BERT4ETH *w.r.t.* two strategies: fixed-training and fine-tuning. For fixed-training, the pre-trained model is used a feature extractor to generate representations, followed by the individual training of a MLP for classification. For fine-tuning, the model is trained with a cascaded MLP together. Each experiment is repeated five times and the best F_1 score is reported with the threshold set to 0.3.

Table 3: Comparison for phishing detection w/ fixed training.

Method	Precision	Recall	F_1
DeepWalk	0.2293	0.1734	0.1974
Trans2Vec	0.1636	0.1432	0.1527
Diff2Vec	0.2184	0.2000	0.2088
Role2Vec	0.2520	0.2295	0.2402
GCN	0.3181	0.2180	0.2587
GSAGE	0.3023	0.2317	0.2623
GAT	0.3264	0.2581	0.2883
BERT4ETH	0.5483	0.4670	0.5044
BERT4ETH†	0.5826	**0.5012**	**0.5388**
BERT4ETH§	**0.5858**	0.4695	0.5212

Table 4: Comparison for phishing account detection w/ fine-tuning. BERT4ETH w/o pre-training equals to Transformer.

Method	Precision	Recall	F_1
BERT4ETH	0.7153	0.5984	0.6516
BERT4ETH†	**0.7421**	**0.6125**	**0.6711**
BERT4ETH§	0.7162	0.6107	0.6593
w/o pre-training			
BERT4ETH	0.5114	0.3845	0.4389
BERT4ETH†	**0.5287**	**0.4129**	**0.4637**
BERT4ETH§	0.5110	0.4058	0.4524

Table 3 summarizes the results of the fixed-training strategy. From the table we can observe that: 1) GNN-based methods, especially GAT, outperform DeepWalk-based methods. 2) The basic BERT4ETH achieves a significant performance boost compared to other baselines: the performance gap of F_1 is up to **21.61** AP compared to GNNs, and **36.94** AP compared to the original BERT with F_1 of 13.50 (Figure 3). The performance boost mainly comes from a better pre-training process, incorporating fine-grained sequential and transaction information, as well as the superior modeling ability of Transformer. 3) By applying two advanced features, BERT4ETH† and BERT4ETH§ further introduce **3.44** and **1.68** AP gain of F_1, respectively. The comparison of account representations generated by several baselines, as visualized in Figure 5, show that phishing account representations generated by BERT4ETH are more dense and separable.

Table 4 presents the results after fine-tuning, with graph-based methods omitted as they perform worse and some of them cannot be fine-tuned. The first three rows of Table 4 show the results of fine-tuning the pre-trained BERT4ETH models, showing a substantial improvement in performance compared to Table 3. This indicates the effectiveness of fine-tuning. The last three rows are results of ablating the pre-training. Obviously, the performance largely decreases, suggesting the importance of pre-training. Among these results, BERT4ETH† achieves the best performance.

5.3.2 ***De-anonymization:*** For ENS dataset, we construct a candidate set including ENS and normal EOAs (595,373 in total), which is shared by all the ground-truth pairs. For Tornado dataset, we use ground-truth pairs collected from 0.1 ETH and 1 ETH coin-mixers. Each ground-truth pair consisting of a deposit EOA and a withdraw EOA, we construct a candidate set including EOAs that deposited Ether to the mixers prior to the withdrawal time. The withdraw EOA is used to query the deposit EOA within the candidate set. BERT4ETH cannot be fine-tuned for this task due to the limited number of ground-truth pairs, with 288 pairs for the ENS dataset and 182 pairs for the Tornado dataset.

Table 5: Comparison for de-anonymization on the ENS dataset (%).

Method	HR@1	@3	@5	@10	@100	@1000
DeepWalk	2.43	5.21	5.56	7.64	15.28	41.32
Trans2Vec	6.60	8.33	9.03	11.46	19.44	24.65
Diff2Vec	3.82	5.90	6.25	9.03	22.22	43.75
Role2Vec	4.17	5.56	6.25	7.30	17.71	35.07
GCN	1.74	2.78	3.13	4.51	11.46	31.94
GSAGE	5.90	9.03	10.07	12.85	26.39	55.90
GAT	3.13	4.51	5.21	6.25	18.40	47.57
BERT4ETH	16.32	23.26	28.47	32.64	47.92	65.28
BERT4ETH†	**20.14**	**27.43**	**31.60**	**36.11**	**51.39**	**68.75**
BERT4ETH§	18.40	25.69	29.86	34.03	50.00	67.71

Table 6: Comparison for de-anonymization on the Tornado dataset.

Mixer	0.1 ETH			1 ETH		
	# Candidate		# Pair	# Candidate		# Pair
Statistic	405.2		102	263.7		80
Method	**Rank↓**	**HR@1**	**HR@3**	**Rank↓**	**HR@1**	**HR@3**
DeepWalk	112.64	17.65%	20.59%	63.59	26.25%	37.50%
Trans2Vec	100.69	18.63%	28.43%	70.57	26.25%	35.00%
Diff2Vec	82.99	26.47%	37.25%	54.00	35.00%	43.75%
Role2Vec	100.99	21.57%	30.39%	64.65	15.00%	31.25%
GCN	153.75	13.73%	17.65%	87.01	16.25%	25.00%
GSAGE	89.46	30.39%	37.25%	58.01	35.00%	48.75%
GAT	91.25	21.57%	27.45%	60.42	25.00%	38.75%
BERT4ETH	67.96	40.20%	56.86%	42.26	45.00%	62.50%
BERT4ETH†	**62.56**	**51.96%**	**58.64%**	**37.34**	**53.75%**	**66.25%**
BERT4ETH§	67.17	39.22%	55.88%	41.66	46.25%	62.50%

Table 5 presents the comparison results for the ENS dataset. Among the DeepWalk-based methods, Diff2Vec demonstrates the best performance. Among GNNs, GraphSAGE outperforms both GCN and GAT by a large margin, which can be attributed to the fact that homogeneous GNNs can introduce a large amount of noise in the multi-hop aggregation, which de-anonymization is susceptible to, especially given that Ethereum account nodes that are highly likely to be linked to popular account nodes. GraphSAGE is more resistant to noise [14] because of its skip-connection mechanism. Notably, the basic BERT4ETH can exactly de-anonymize 16.32% of account pairs, offering a significant improvement of **9.7** AP gain on HR@1. Additionally, the in/out separation strategy further brings a considerable improvement upon the basic BERT4ETH.

Table 9 presents the results for the Tornado dataset, where Rank is the average rank of the deposit account within the candidate set. The size of candidate set varies due to the unique withdrawal time. Similarly, BERT4ETH significantly outperforms its competitors. The results show that it can even exact de-anonymize **40.2%** of testing pairs for 0.1 ETH mixer and **45.0%** of for 1 ETH mixer.

Table 7: Ablation study for phishing detection w/ fixed-training.

Method	Precision	Recall	F_1
BERT4ETH	0.5483	0.4670	0.5044
w/o de-duplication	0.4661	0.3289	0.3857
w/o 80% masking	0.4177	0.2431	0.3073
w/o freq. sampling	0.4554	0.3266	0.3804
w/o batch-sharing	0.5034	0.3661	0.4239
w/o tranx. features	0.5076	0.3607	0.4217

Table 8: Ablation study for de-anonymization on ENS dataset (%).

Method	HR@1	@3	@5	@10	@100	@1000
BERT4ETH	16.32	23.26	28.47	32.64	47.92	65.28
w/o de-duplication	18.06	26.39	30.21	32.29	46.18	59.38
w/o 80% masking	16.67	23.96	27.43	31.60	41.33	52.78
w/o freq. sampling	4.52	9.03	10.07	11.11	27.78	48.61
w/o batch-sharing	0.69	1.74	3.82	5.90	21.88	43.40
w/o tranx. features	15.28	21.53	26.74	27.09	42.71	62.85

5.4 Ablation Study

We investigate the impact of all the proposed strategies by ablating five key elements of BERT4ETH, *i.e.*, transaction de-duplication, high masking ratio, frequent negative sampling, intra-batch sharing and transaction features.

Table 7 presents the results of the ablation study conducted on the phishing account detection task with fixed training. First, we observe that removing each one of them results in a noticeable performance decline. Second, we observe that switching masking ratio to BERT's original setting (15%) lead to the largest performance decrease, indicating that repetitiveness can largely hurt the effectiveness of pre-training for the phishing detection task.

On the contrary, the conclusion drawn from the ablation study on the de-anonymization task is entirely different. Ablating repetitiveness reduction strategies (de-duplication & 80% masking) actually lead to a slight performance increase when $k \leq 5$, suggesting that de-anonymization task is not particularly susceptible to high repetitiveness. Another noteworthy finding is that two skew alleviation strategies are crucial for this task. It is worth mention that skew alleviation is important especially when the masking ratio is high. This is because high-frequency addresses are more likely be selected as the positive addresses, which can cause the whole sequence representation to be closer to them, making the representations indistinguishable.

Table 9: Case study of hot-to-cold query for de-anonymization.

	Case 1		Case 2	
Query type	**cold-to-hot**	**hot-to-cold**	**cold-to-hot**	**hot-to-cold**
Diff2Vec	8th	47,641st	10th	49,059th
BERT4ETH	8th	20th	2th	54th

5.5 Case Study

We identify the case of hot-to-cold query for de-anonymization where graph-based methods may fail. In this case, the cold account has much less number of transactions than the hot account. Take

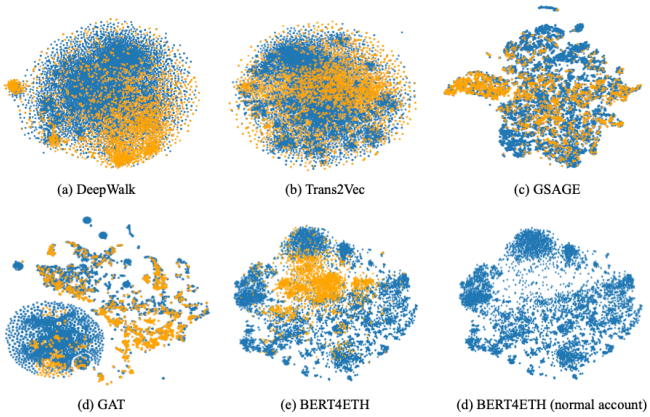

Figure 5: T-SNE visualization of phishing (orange) and normal (blue) accounts for several competitors.

`Case 1` for example, when using the cold account to query the hot account, both BERT4ETH and Diff2Vec rank the hot account at 8th/595,396. However, when using the hot account to query the cold account, BERT4ETH ranks 20, and Diff2Vec ranks 47,641. The reason is that graph-based methods can introduce a large amount of noise when a node's neighborhood is large. In contrast, BERT4ETH preserves the first order of neighborhood with transaction-level sequential information, and emphasizes important information via self-attention, making it still effective for hot-to-cold query.

`Case 1`:
Cold EOA: 0x92fa836a964fd5544545b18285475f715acb5576 (7)
Hot EOA: 0xf199b022e3edab7e8ac6e214fad1bdfd31703766 (922)

`Case 2`:
Cold EOA: 0x53c5438e8c825ba574865b4c1ccb34e74b3affdf (5)
Hot EOA: 0xd9ceb2bb4b8324d36c284799eb00c7cc19a0f618 (143)

6 CONCLUSION

We present BERT4ETH, a pre-trained Transformer that offers a universal solution for fraud detection tasks on Ethereum. BERT4ETH features the superb modeling ability of the Transformer and incorporates three effective strategies to tackle the challenges of pre-training a BERT model for Ethereum. These strategies, namely repetitiveness reduction, skew alleviation and heterogeneity modeling, result in substantial improvements and operate cohesively and harmoniously. The significant improvements achieved on phishing account detection and de-anonymization tasks suggest that BERT4ETH is well suited for practical applications.

7 ACKNOWLEDGEMENT

This research was supported by the National Research Foundation, Singapore under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore. Authors of Georgia Tech acknowledge the partial support by the US National Science Foundation under Grants NSF 1564097, NSF 2026945, NSF 2038029, a CISCO grant and an IBM Faculty Award.

REFERENCES

[1] N. K. Ahmed, R. Rossi, J. B. Lee, T. L. Willke, R. Zhou, X. Kong, and H. Eldardiry. Learning role-based graph embeddings. *arXiv preprint arXiv:1802.02896*, 2018.

[2] F. Béres, I. A. Seres, A. A. Benczúr, and M. Quintyne-Collins. Blockchain is watching you: Profiling and deanonymizing ethereum users. In *2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pages 69–78. IEEE, 2021.

[3] S. Bian, Z. Deng, F. Li, W. Monroe, P. Shi, Z. Sun, W. Wu, S. Wang, W. Y. Wang, A. Yuan, et al. Icorating: A deep-learning system for scam ico identification. *arXiv preprint arXiv:1803.03670*, 2018.

[4] W. Chen, X. Guo, Z. Chen, Z. Zheng, and Y. Lu. Phishing scam detection on ethereum: Towards financial security for blockchain ecosystem. In *IJCAI*, pages 4506–4512, 2020.

[5] W. Chen, T. Zhang, Z. Chen, Z. Zheng, and Y. Lu. Traveling the token world: A graph analysis of ethereum erc20 token ecosystem. In *Proceedings of The Web Conference 2020*, pages 1411–1421, 2020.

[6] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, and Y. Zhou. Detecting ponzi schemes on ethereum: Towards healthier blockchain technology. In *Proceedings of the 2018 world wide web conference*, pages 1409–1418, 2018.

[7] CoinDesk. Uniswap user loses $8m worth of ether in phishing attack, 2022. https://www.coindesk.com/tech/2022/07/12/uniswap-user-loses-8m-worth-of-ether-in-phishing-attack.

[8] T. R. Cryptocurrency. 100s of eth stolen after bee token ico email list hacked, 2022. https://theripplecryptocurrency.com/bee-token-scam.

[9] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. *arXiv preprint arXiv:1904.05234*, 2019.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[11] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[12] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] S. Hu, X. Zhang, J. Zhou, S. Ji, et al. Turbo: Fraud detection in deposit-free leasing service via real-time behavior network mining. In *ICDE*, 2021.

[15] S. Hu, Z. Zhang, S. Lu, B. He, and Z. Li. Sequence-based target coin prediction for cryptocurrency pump-and-dump. *SIGMOD*, 2023.

[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[17] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[18] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[19] X. T. Lee, A. Khan, S. Sen Gupta, Y. H. Ong, and X. Liu. Measurements, analyses, and insights on the entire ethereum blockchain network. In *Proceedings of The Web Conference 2020*, pages 155–166, 2020.

[20] S. Li, G. Gou, C. Liu, C. Hou, Z. Li, and G. Xiong. Ttagn: Temporal transaction aggregation graph network for ethereum phishing scams detection. In *Proceedings of the ACM Web Conference 2022*, pages 661–669, 2022.

[21] D. Lin, J. Wu, Q. Yuan, and Z. Zheng. Modeling and understanding ethereum transaction records via a complex network approach. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(11):2737–2741, 2020.

[22] D. Lin, J. Wu, Q. Yuan, and Z. Zheng. T-edge: Temporal weighted multidigraph embedding for ethereum transaction network analysis. *Frontiers in Physics*, 8:204, 2020.

[23] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[24] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[25] B. Rozemberczki and R. Sarkar. Fast sequence-based embedding with diffusion graphs. In *International Workshop on Complex Networks*, pages 99–107. Springer, 2018.

[26] J. Shen, J. Zhou, Y. Xie, S. Yu, and Q. Xuan. Identity inference on blockchain using graph neural network. In *International Conference on Blockchain and Trustworthy Systems*, pages 3–17. Springer, 2021.

[27] Y. Tang, C. Xu, C. Zhang, Y. Wu, and L. Zhu. Analysis of address linkability in tornado cash on ethereum. In *China Cyber Security Annual Conference*, pages 39–50. Springer, 2021.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, et al. Attention is all you need. *arXiv:1706.03762*, 2017.

[29] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[30] F. Victor and A. M. Weintraud. Detecting and quantifying wash trading on decentralized cryptocurrency exchanges. In *Proceedings of the Web Conference 2021*, pages 23–32, 2021.

[31] J. Wu, Q. Yuan, D. Lin, W. You, W. Chen, C. Chen, and Z. Zheng. Who are the phishers? phishing scam detection on ethereum via network embedding. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[32] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[33] L. Zhao, S. Sen Gupta, A. Khan, and R. Luo. Temporal analysis of the entire ethereum blockchain network. In *Proceedings of the Web Conference 2021*, pages 2258–2269, 2021.

[34] J. Zhou, C. Hu, J. Chi, J. Wu, M. Shen, and Q. Xuan. Behavior-aware account de-anonymization on ethereum interaction graph. *arXiv preprint arXiv:2203.09360*, 2022.